username_lab01_stub

January 7, 2024

1 Lab 01 - Linear Algebra and Numpy

Welcome to the first lab in CSC4601/5601! If you can edit this you probably have a working instance of jupyter notebook (either locally or on Rosie). If you are looking at this as an pdf, maybe you still need to get an instance of Jupyter running. Please follow the neccessary steps in Experiment 1.

2 Experiment 1

In this experiment you will be making sure that you can connect to Rosie and run an interactive session (jupyter notebook session). You should have an account on Rosie from a previous class, but if you have a new account (or haven't used yours in so long that your password has expired) you will have to reset your password. Ask your instructor or the Rosie administrator for the default password. To do this you will have to access the terminal on Rosie - meaning you will have to ssh in. Once you have reset your password, you will be able to access Rosie's web portal and initiate interactive session from there. The following steps and sections will give you what you need to start.

2.1 Local Alternative to Rosie

For some labs you may want to run everything locally. Although you lack Rosie's amazing computing power, this can be easier and more flexible for development, small jobs, etc. One method to set up an appropriate environment on your Windows laptop is:

1. Install Anaconda
2. Open an Anaconda Prompt
3. `conda create -n csc4601 python jupyter numpy ipython scipy matplotlib pandoc`
4. `conda activate csc4601`
5. `copy username_lab01_stub.ipynb yourUsername_lab01.ipynb`
6. `jupyter notebook`
7. ctrl-c from the console to stop the jupyter server

You should also install MiKTeX. Among other things, it is required to use the File | Save and Export Notebook as… | PDF option if the notebook contains LaTeX markup, as they often do when including equations (such as in this file).

2.2 Accessing Rosie

An objective of this class is to give you some more experinece using remote resources and Rosie is a great resource to have. Our admin is Dr. Retert. Please refer to Rosie's webpage as a first step in finding solutions to issues you may be having. Your instructor is also a good resource if troubleshooting is required.

2.2.1 SSH Client

If you are on Windows, you will have to download and install an ssh client. A commonly used and free client is Putty. Please follow the link and install Putty on your machine.

2.2.2 On network or off

If you are doing these steps off-campus, you will need to use MSOE's VPN to access the network that ROSIE is on. To do this you can follow the written instruction on Rosie User Guide – Network Access.

2.2.3 Starting an Interactive session

Once you have access to Rosie's network and you have a username and current password (done through the SSH client), you can complete the steps for starting an interactive session. You should access ROSIE's web portal and start a jupyter notebook session to run (and complete) this notebook.

3 Experiment 2 - Structuing your Data and Feature Matrices / Slicing

In this experiment you will refamiliarize yourself with python/numpy and use some of the common data manipulation techniques that you will need for the rest of the class.

3.1 What is Numpy?

- Matrix library
- Memory-efficient data structures – arrays
  - Used in scikit-learn, matplotlib, and others
- Expressive API for indexing and operations
- Time-efficient algorithms
  - Calls C and Fortran libraries where possible

3.2 How Do I Import Libraries into my Jupyter Notebook working kernel?

- The following bit of code can be used to import libraries. The world is your oyster!

[ ]:
```
import numpy as np
import scipy
import scipy.stats as stats
import matplotlib.pyplot as plt
# datapath = '/data/cs3400/datasets/IRIS.csv' # if running on Rosie
```

```
datapath = './IRIS.csv' # if running locally
```

3.3 How to read in files, organize data, and plot some features!

In the first step you will read the IRIS.csv file that you are given (which is also on our class's datashare on ROSIE) and put the features into a matrix. In machine learning the standard for organizing matrices is always observations in rows, and features that describe the observations as columns. Read in the data file and assign the data to a numpy matrix.

1. Use the function numpy.loadtxt.
   - You will want to use the proper delimiter for the file you have.
   - Make sure that you skip any text rows, numpy matrices can only be a single datatype.
   - Depending on the dataset you may need to specify what columns you want to use.
   - With your data matrix you should explore the data a bit.
2. Use `data.shape` to find your dimensions
3. Plot the first two features your data using matplotlib. Label all of your axes and use legends!
   1. Make a figure with a line plot
   2. Make another figure with a scatter plot
   3. Make a third figure displaying both the same line and scatter plots.
4. Print all of the feature values for the 150th observation in your dataset.
5. Select observations 49-52 from your dataset and print them to the notebook.
6. Select all of the entries in your dataset that have their first feature <= 5 and print the first 5 results. (hint: do this in multiple steps. First make a boolean mask of your matrix)
7. Calculate the median, standard deviation, and mode of the entries selected in the previous step. (Hint 1: these should be done column by column. Hint 2: Don't forget about other packages like scipy!)

3.3.1 1) Load the IRIS.csv file into a numpy matrix named `data`

[ ]:

3.3.2 2) Display its dimensions (`data.shape`)

[ ]:

3.3.3 Plot the first two features of your data using matplotlib. Label all of your axes and use legends!

3.3.4 3-A) Make a line plot of the first two dimensions using matplotlib

You may find the beginner's cheatsheet from https://matplotlib.org/ useful.

[ ]:

3.3.5 3-B) Make a scatter plot of the first two dimensions using matplotlib

Hint: One way to turn the lines connecting points off is to set the linestyle in plot() to 'none'.

[ ]:

3.3.6 3-C) Make a third figure displaying 2 subplots: both the above line and scatter plots

One way to do this is with the subplot() function, which lets you have a grid of subplot axes within a single figure.

[ ]:

4) Print all of the feature values for the 150th observation in your dataset. Since this is the final row in the dataset, you may also index it as -1. Remember that negative indexes work backward from one past the final element.

[ ]:

3.3.7 5) Select observations 49-52 from your dataset and print them to the notebook.

Since there are 4 integers between 49 and 52 inclusive, this means 4 observations. Make sure you index the array correctly

[ ]:

3.3.8 6) Select all of the entries in your dataset that have their first feature <= 5 and print the first 5 results (Hint: Do this in multiple steps. First make a boolean mask of your matrix)

[ ]:

3.3.9 7) Calculate the median, standard deviation, and mode of the entries selected in the previous step.

Hints: 1. You may need NumPy for some of these and SciPy for others. 2. These should be done column by column.

[ ]:

4 Experiment 3 - Linear Algebra in Numpy

In this experiment you will be performing a number of linear algebra operations in your jupyter notebook. Check out the linalg module of numpy!

We have started by creating a few vectors and matrices for you.

[ ]:

You will: 1. Create a few more numpy vectors and matrices 2. Print the number of dimensions each of your numpy vectors and matrices 3. Print the shape (length and dimension) of each of your numpy vectors and matrices 4. Print the datatype used in each of your numpy vectors and matrices 5. Try to compute a dot product on two matrices of with disagreeable dimensions 6. Compute a dot product on two matrices with agreeable dimensions 7. Try to compute element-wise addition on two matrices with disagreeable dimensions 8. Compute an element-wise addition on two matrices with

agreeable dimensions 9. Compute the norm (distance) between a vector and itself 10. Compute the norm (distance) between two different vectors 11. Apply a set of linear coefficients to a matrix of observations.

4.0.1 1) Create numpy vectors and matrices (we have done a few for you)

[ ]:

4.0.2 2) Print the number of dimensions each of your numpy vectors and matrices

[ ]:

4.0.3 3. Print the sahpe (length and dimension) of each of your numpy vectors and matrices

[ ]:

4.0.4 4) Print the datatype used in each of your numpy vectors and matrices

[ ]:

4.0.5 5) Try to compute a dot product on two matrices of with disagreeable dimensions

[ ]:

4.0.6 6) Compute a dot product on two matrices with agreeable dimensions

[ ]:

4.0.7 7) Try to compute element-wise addition on two matrices with disagreeable dimensions

[ ]:

4.0.8 8) Compute an element-wise addition on two matrices with agreeable dimensions

[ ]:

4.0.9 9) Compute the norm (distance) between a vector and itself

[ ]:

4.0.10 10) Compute the norm (distance) between two different vectors

[ ]:

4.0.11 11) Apply a set of linear coefficients to a matrix of observations.

From your problem set you can see the form of this model:

$$y = \beta_0^{12} + \beta_1 x_1 + \beta_2 x_2 + \beta_3 x_3$$

which can also be represented in vector notation as:

$$y = x^T$$

Use the vectors that you created in problem 5 of problem set 1 and evaluate it here. Evaluate it twice, once using matrix multiplication and once with dot products.

Hints: 1. To create a random vector of length 4, which might be useful for x and β, use np.random.rand(4) 2. Use np.matmul() for matrix multiplication. (The * operator is *element-wise* multiplication in NumPy. It works with broadcasting and will give you unxpected results if you're trying to do matrix multiplication.) 3. One way to make a 1-D vector into a 2-D matrix is using the expand_dims() function.

[ ]:

5 Bonus Material: Additional Indexing Topics

Before considering the following indexing procedures, think about the following question. Can I index a vector (nx1) using a matrix (nxm)? What would happen if I try?

[ ]:
```
X = np.random.randint(10, size=(10, 3))
y = np.expand_dims(np.array([1, 0, 1, 1, 0, 0, 2, 2, 1, 0], dtype=np.
  ↪int32),axis=1)
```

Think of the above matrix, X, as a feature matrix (10x3) and the above vector, y, as a response vector/matrix (10x1). How can I index and get the first index of X or y?

[ ]:
```
y[0,0]
```

[ ]:
```
X[0,0]
```

What if I want multiple elements from this array that are not sequential? Such as element 0 and element 7?

[ ]:
```
print(y[0,0])
print(y[7,0])
```

Pretty straighforward, eh? Can I do this in one go?

[ ]:
```
print(y[[0,7],[0,0]])
```

Not too shabby! Now, is there anything preventing me from re-indexing the same element? Let's try!

[ ]:
```
print(y[[7,7],[0,0]])
```

woah

Finally, lets take this to a ridiculous conclusion… What happens if I supply more index calls (as a matrix) than the variable has in shape?

[ ]:
```
print(y[[7,7,7,7,7,7,7,7,7,7,7,7,7,7,7,7],[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]])
```